2. Web statique et formats de données

M1 Outils de l'Internet
lundi 27 septembre 2010

victor.poupet@lif.univ-mrs.fr

Wold Wide Web

ENQUIRE

- Années 80, au CERN
- Tim Berners-Lee propose un système d'informations réparties sur le réseau, reliées par des liens hyper-textes
- Écrit en PASCAL
- Utilisation limitée aux milieux scientifiques

ENQUIRE

- Base de données
- Liens bi-directionnels
- Edition directe sur le serveur (type wiki)
- Facile à écrire, ajouter des liens, des fichiers, etc.

World Wide Web

- 1989 évolution du projet
- Inspiration puisée dans le langage SGML
- Premier navigateur : *WorldWideWeb*
- Premier serveur, et premières pages...

WorldWideWeb
Info
Navigate
Document
Find
Edit
Links
Style
Print... p
Page layout...
Windows
Services
Hide h
Quit q

Links
Mark all A
Mark selection M
Link to marked L
Link to New N
Unlink Z
Link to file...
Help

Tim's Home Page

My home page

altas

The World-Wide Web Virtual Library: Subject Catalogue

The WWW Virtual Library

This is a c
type ., an

Mail to ma
to add po
administra

See also

Aeronaut

Agricultur

Anthropology

Archaeology

Asian Studies

Astronomy and

Bio Sciences

Separate list.

Separate list .

High-Energy Physics Information

CERN Welcome

CERN

European Laboratory for Part

Geneva, Switzerlan

About the Laboratory:

- on CERN info ,
- General information, divisions, groups and activities , scientific committees

CERN Experiments

Experiments

WWW Support for Experiments

ALEPH	LEP experiment
ALICE	A Large Ion Collider Ex LHC
ATLAS	A Toroidal LHC Appara
CHORUS	WA95 - Neutrino oscill CERN
CMS	Compact Muon Solenc

Atlas

Caractéristiques du WWW

- 3 points forts du web :
  - Le format des fichiers (dérivé de SGML)
  - le premier protocole HTTP
  - l'espace d'adressage des hyper-liens

Markup Languages

SGML

Standard Generalized Markup Language (iso 8879:1986)

- Origine : IBM GML (années 60, Goldfarb, Mosher, Lorie)
- Documentation de gros projets
- Facile à traiter automatiquement
- Le format doit rester lisible pendant des dizaines d'années

SGML

- Un méta-langage pour définir des langages d'annotation (spécifiés par une DTD)
- Annotation par balise :
  `<balise nom="blop"> ... </balise>`
- Les annotations donnent la structure du contenu, pas la façon dont il doit être traîté

Validité

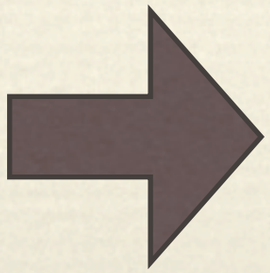
- Document bien formé : les balises sont bien imbriquées et fermées
- Document valide : correspond à la DTD

HTML

- Syntaxe inspirée de SGML
- Apparaît en 1991
- Devient un format SGML en juin 1993
- Effort de normalisation : IETF puis W3C

Versions

- 1991 : HTML Tags
- 1992 : HTML DTD
- 1995 : HTML 3.0 IETF draft (et CSS)
- Evolution jusqu'à HTML 4.01 en 1997
- 2008 : HTML 5 (working draft)
- En parallèle depuis 2000 : XHTML



La sytaxe est beaucoup plus simple...

Document HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<HTML>

  <HEAD>
    <TITLE>Titre de la page</TITLE>
  </HEAD>

  <BODY>
    <P>Voici un paragraphe de texte.
    <P>En voici un second.
  </BODY>

</HTML>
```

DOCTYPE

- Tout document SGML valide contient une déclaration de type de document
- Annonce que ce qui suit est du HTML
- Précise la version de la norme utilisée

<HTML>

- La balise <HTML> encadre le document
- Elle contient les méta-informations du document <HEAD> suivies du corps <BODY>

Comme toute balise SGML, elle accepte des attributs (`lang`, `version`, etc.)

<HEAD>

- Contient les méta-informations :
  - Titre <TITLE>
  - Informations d'indexation <META>
  - etc...

<BODY>

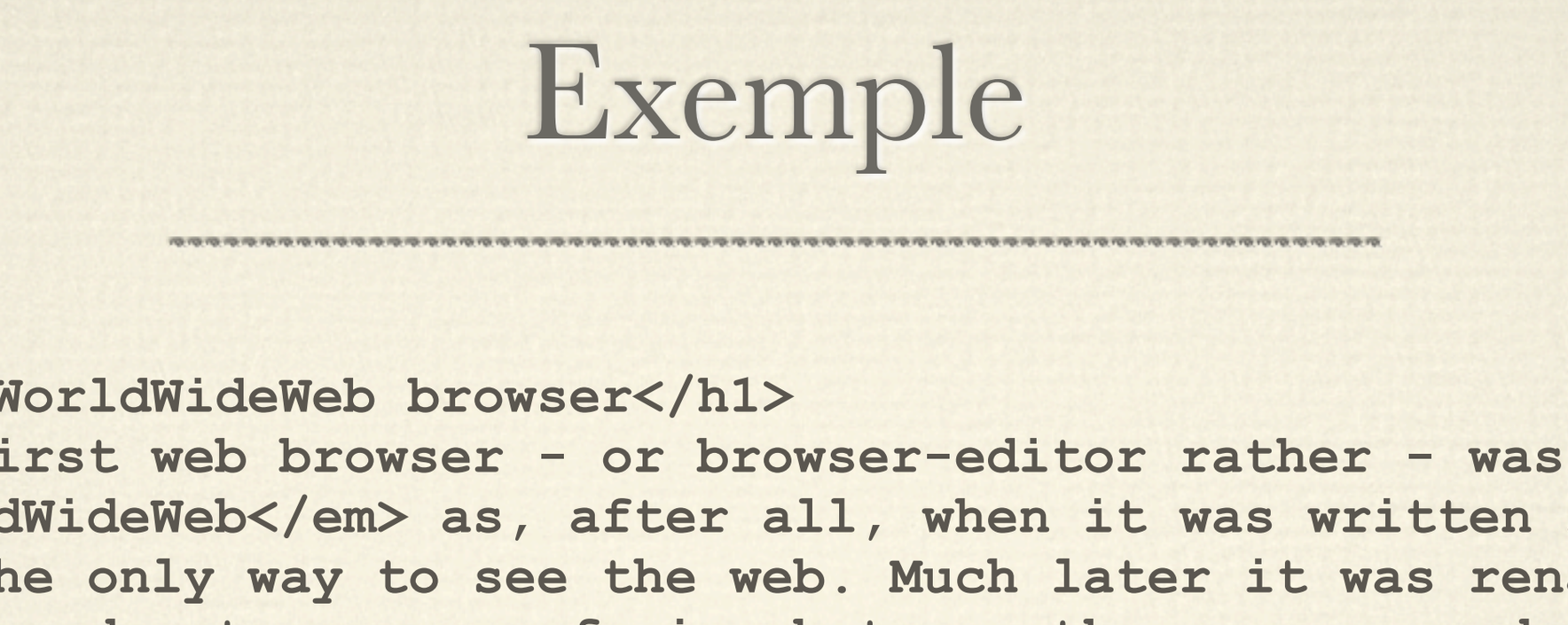
- Le corps même du document
- Du texte brut avec des balises
- Des entités SGML comme par exemple ` `, `&amp`, `&eacute`, etc.
- SGML est insensible à la casse des balises

Mise en forme du texte

- Les informations sont sémantiques ou visuelles :
  - <H1>, <Hn>, <P>, <EM>, <STRONG>, <CITE>, <BR>, etc.
  - <TT>, <I>, <HR>, <SMALL>, <BIG>, <FONT>, etc.

Exemple

```
<h1>The WorldWideWeb browser</h1>
<p>The first web browser - or browser-editor rather - was called
<em>WorldWideWeb</em> as, after all, when it was written in 1990
it was the only way to see the web. Much later it was renamed
Nexus in order to save confusion between the program and the
abstract information space (which is now spelled <em>World Wide
Web</em> with spaces).</p>
```



The WorldWideWeb browser

The first web browser - or browser-editor rather - was called *WorldWideWeb* as, after all, when it was written in 1990 it was the only way to see the web. Much later it was renamed Nexus in order to save confusion between the program and the abstract information space (which is now spelled *World Wide Web* with spaces).

http://www.w3.org/People/Berners-Lee/WorldWideWeb.html

Listes et tables

- Listes à puces <UL><LI>, numérotées <OL> ou encore à mots-clés : <DL>, <DT>, <DD>.
- Tableaux avec <TABLE>, <TH>, <TR> et <TD>

<A href="...">

- Les liens hypertexte en HTML
- Ancre interne au document :
  `<A NAME="ancre">texte</a>`
- Hyper-lien vers une ancre :
  `<A HREF="#ancre">go</A>`
- Un hyper-lien quelconque :
  `<A HREF="page.html#ancre">lien</A>`

Un autre exemple

```
<h2><a name="a4267"></a><a href="/browse/authors/
b#a4267">Bonaparte, Napoléon, 1769-1821</a></h2>
<ul>
<li class="pgdbetext"><a href="/etext/12230">Oeuvres de
Napoléon Bonaparte, Tome I.</a> (French)</li>
<li class="pgdbetext"><a href="/etext/12782">Oeuvres de
Napoléon Bonaparte, Tome II.</a> (French)</li>
<li class="pgdbetext"><a href="/etext/12893">Oeuvres de
Napoléon Bonaparte, Tome III.</a> (French)</li>
<li class="pgdbetext"><a href="/etext/13192">Oeuvres de
Napoléon Bonaparte, Tome IV.</a> (French)</li>
<li class="pgdbetext"><a href="/etext/13475">Oeuvres de
Napoléon Bonaparte, Tome V.</a> (French)</li>
<li class="pgdbetext"><a href="/etext/19700">Tendresses
impériales</a> (French)</li>
</ul>
```

Un autre exemple

Bonaparte, Napoléon, 1769-1821

- Oeuvres de Napoléon Bonaparte, Tome I. (French)
- Oeuvres de Napoléon Bonaparte, Tome II. (French)
- Oeuvres de Napoléon Bonaparte, Tome III. (French)
- Oeuvres de Napoléon Bonaparte, Tome IV. (French)
- Oeuvres de Napoléon Bonaparte, Tome V. (French)
- Tendresses impériales (French)

http://www.gutenberg.org/browse/languages/fr

<IMG>, <OBJECT>, <APPLET>

- Tout est plus joli avec des images :
  `<IMG SRC="kitty.jpg" ALT="mignon">`
- Inclusion de contenus multimedia embarqués `<OBJECT>`, `<APPLET>`.

Attention à l'accessibilité !

<FORM>

- Un début d'interactivité : les formulaires
- <FORM METHOD="..." ACTION="...">
- <LABEL ...>,<INPUT TYPE="text" id="...">,<BUTTON ...>, <SELECT ...>
- Envoyé vers une URL (mail ou web principalement)

Troisième exemple

```
<form METHOD="GET" action="/cgi-bin/mailformclub"
onsubmit="return Fclub_valide(this)" name="Fclub">
<blockquote><blockquote><p>
  Prénom : <input TYPE="TEXT" NAME="PRENOM" SIZE="20">
  Nom : <input TYPE="TEXT" NAME="NOM" SIZE="20">
  <br><br>
  e-Mail : <input TYPE="TEXT" NAME="MAIL" SIZE="40">
  <br><br>
  Message :
  <br><textarea rows="7" NAME="MESSAGE" cols="50">
  </textarea> <br><br>
<input TYPE="SUBMIT" NAME="B1" VALUE="Envoyer"> </p>
</blockquote></blockquote>
</form>
```

http://senat.fr/evenement/clubsenat/formulaire.html

Troisième exemple

Si vous souhaitez obtenir des informations complémentaires sur le Club.Sénat.fr, merci de bien vouloir remplir le formulaire suivant :

Prénom : Nom :

e-Mail :

Message :

Envoyer

http://senat.fr/evenement/clubsenat/formulaire.html

Le protocole HTTP

Hypertext Transfer Protocol

- Développé pour obtenir des documents liés par liens hypertextes (fin des anneés 80)
- Défini et mis à jour par le W3C et l'IETF
- Fonctionnement client/serveur : une requête suivie d'une réponse
- Sans état (*stateless*)

Versions

- 0.9 - Uniquement l'action GET. Ne précise pas le numéro de version (obsolète)
- 1.0 (mai 96) - Encore couramment utilisé
- 1.1 (97-99) - Version actuelle, connexions persistantes par défaut (RFC 2616)

Exemple simple

[client] GET unepage HTTP/1.0

[serveur] HTTP/1.0 200 OK
Content-Type: text/html

<!DOCTYPE ...>
<HTML>
...

Commandes HTTP

- `GET` (*) demande la transmission d'une ressource (un document)
- `POST` idem, mais le client soumet des données (formulaire)
- `HEAD` (*) demande uniquement les en-têtes
- `PUT` (désuet) mise à jour de ressources
- `DELETE` (désuet) efface une ressource
- `TRACE`, `OPTIONS`, `CONNECT`

Effets de bord

- Les méthodes `GET`, `HEAD`, `TRACE`, `OPTIONS` et `CONNECT` sont "sûres"
- Les méthodes `POST`, `PUT` et `DELETE` peuvent provoquer des modifications du serveur
- `PUT` et `DELETE` devraient être idempotentes

Codes de retour

- `1xx` information
- `2xx` succès, la réponse suit
  `200 OK`, `202 Accepted`, etc.
- `3xx` redirection : la ressource est ailleurs
  `301 Moved Permanently`, `302 Not Found`
- `4xx` erreur de requête
  `400 Bad Request`, `403 Forbidden`, `404 Not Found`
- `5xx` erreur du serveur
  `500 Internal Server Error`, `501 Not Implemented`, `503 Service Not Available`

Headers HTTP (client)

```
Accept: */*
Accept-Language: fr
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Macintosh; U; PPC
Mac OS X; fr) AppleWebKit/418.9.1 (KHTML,
like Gecko) Safari/419.3
Connection: keep-alive
Host: localhost:7777
```

Headers HTTP (serveur)

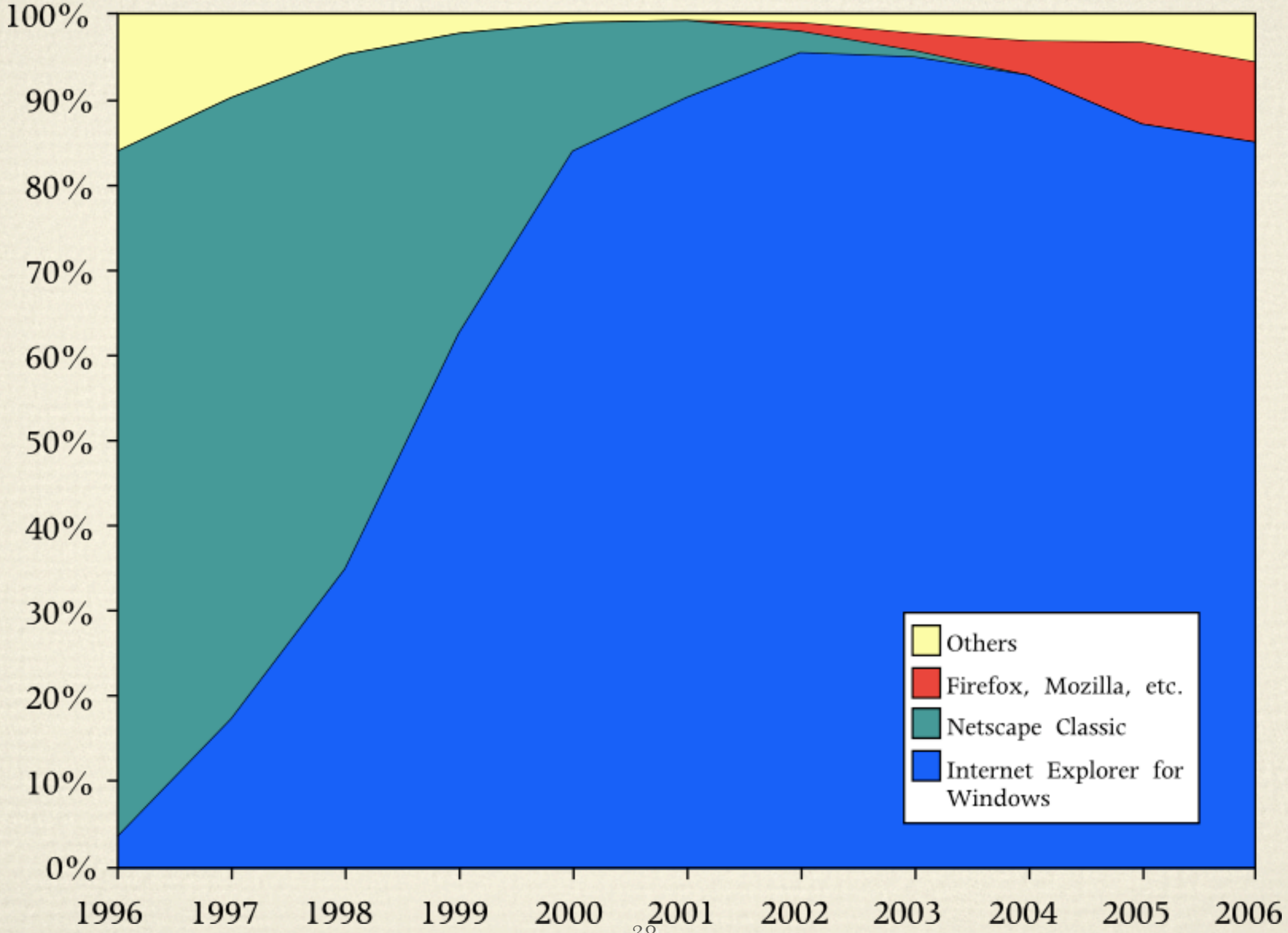
```
Date: Thu, 08 Feb 2007 20:48:45 GMT
Server: Apache
Last-Modified: Thu, 08 Feb 2007 19:20:26 GMT
ETag: "8057fc-10d6f-45cb77fa"
Content-Length: 68975
Content-Type: text/html; charset=iso-8859-1
```

Navigateurs

Navigateur web

- Émet des requêtes vers les serveurs web (ou va chercher le fichier en local)
- Affiche le contenu (HTML, plug-ins)
- Gère l'interaction (formulaires, clics)
- C'est un interprète dédié au web

Browser Wars



Ecrire du HTML

- La réussite de HTML est liée à sa simplicité
- Les navigateurs son très souples
- Possibilité d'écrire toute une page web avec un éditeur pur texte

Générer du HTML

- Le HTML est un format facile à produire automatiquement
- C’est la sortie rêvée pour un programme
- C’est tellement facile qu’on a envie d’en générer à la volée

Parser du HTML

- Par contre écrire un parseur HTML est un véritable cauchemar...
- Ecrire un navigateur, c'est encore pire !

XML

(eXtensible Markup Language)

- Heureusement, voici XML
- Sous-ensemble simplifié de SGML
- Plus strict
- Plus facile à parser
- Développé par le W3C

Les arbres en XML

- XML est un format de texte
- Mais ça décrit une structure arborescente

```
<racine>
   <fils1/>
   <fils2>
      <sous-fils/>
   <fils2/>
   <fils3/>
</racine>
```

Syntaxe

- Comme en SGML on a des balises
- Les noms de balises et d'attributs sont en minuscule
- Les valeurs des attributs sont toujours entre guillemets
- Toute balise ouverte doit être fermée
- On a la forme raccourcie `<blop/>`

Syntaxe

- On commence toujours par
  `<?xml version="1.0" encoding="UTF-8"?>`
- Tout document possède une unique racine
- Les éléments sont correctement imbriqués
  `<blop>...<blip>...</blip>...</blop>`

Syntaxe

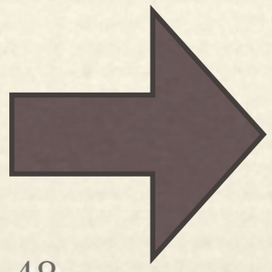
- Comme en SGML on a des entités spéciales : `<`, `&`, `&apos;`, etc.
- Si on en veut d'autres on les définit dans la DTD
- On a aussi des références numériques : `&#65;`, `&#x27;`

Espaces de noms

- Pour mélanger différents vocabulaires XML
- On a un espace de nom principal
  `xmlns="http://www.w3.org/1999/xhtml"`
- Et des espaces secondaires (nommés) :
  `xmlns:svg="http://www.w3.org/2000/svg"`
- On nomme les entités :
  `<svg:path>`

Des grammaires

- Plusieurs options : DTD, XML Schema, etc.
  Rq : les DTD ont des DOCTYPE
- Même rôle que les DTD en SGML
- Spécifient les arbres valides

Document bien formés  Documents valides

XML (bilan)

- Facile à parser
- Facile à générer
- Plus rigoureux
- Sujet à la mode
- Toute la puissance des arbres dans un document texte (mais pas très compact)

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//
DTD PLIST
1.0//EN" "http://www.apple.com/DTDs
PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>
      ApplicationCrashedAfterRelaunch
    </key>
    <integer>0</integer>
  </dict>
</plist>
```

XHTML

- Standard W3C XHTML 1.0
- On prend tout HTML 4.0 avec une syntaxe XML

 <hr/>
- La conversion HTML → XHTML est très simple (et automatisable)

Routage sur le web

Ressources

- Ressource : document, objet, personne, etc.
- Comment identifier une ressource ?
- Comment adresser une ressource ?
- Ressource relative, ressource absolue...

Au commencement...

- Analogie avec le système de fichiers
- Hyper-liens, images, adressage relatif : `src="kitten.png"`, `src="/img/kitten.png"`
- Adressage absolu : `src="http://kitten.org/funky.jpg"`

Généralisation

- Messagerie :
  `href="mailto:guybrush@melee.org"`
- Serveur FTP :
  `href="ftp://ftp.gnu.org/README"`
- Fichier local :
  `src="file:///home/lechuck/bigwhoop.jpg"`

Formalisation

- Plein d'acronymes :
  - URI identifieur de ressource (URL, URN)
  - URL localisation de ressource
  - URN nom de ressource
  - IRI des URI en Unicode plutôt qu'en ASCII
  - XRI extensions de l'OASIS

Plein de RFC

RFC 1630 URLs, informel 1994

RFC 1737 URNs, informel

RFC 1738 URLs, proposition standard

RFC 1808 URLs relatives, proposition

RFC 2141 URN syntaxe

RFC 2396 URI syntaxe

RFC 2717, 2718 URLs en pratique

RFC 3987 IRIs la norme

RFC 3986 URIs syntaxe 2005

Nommer : URN

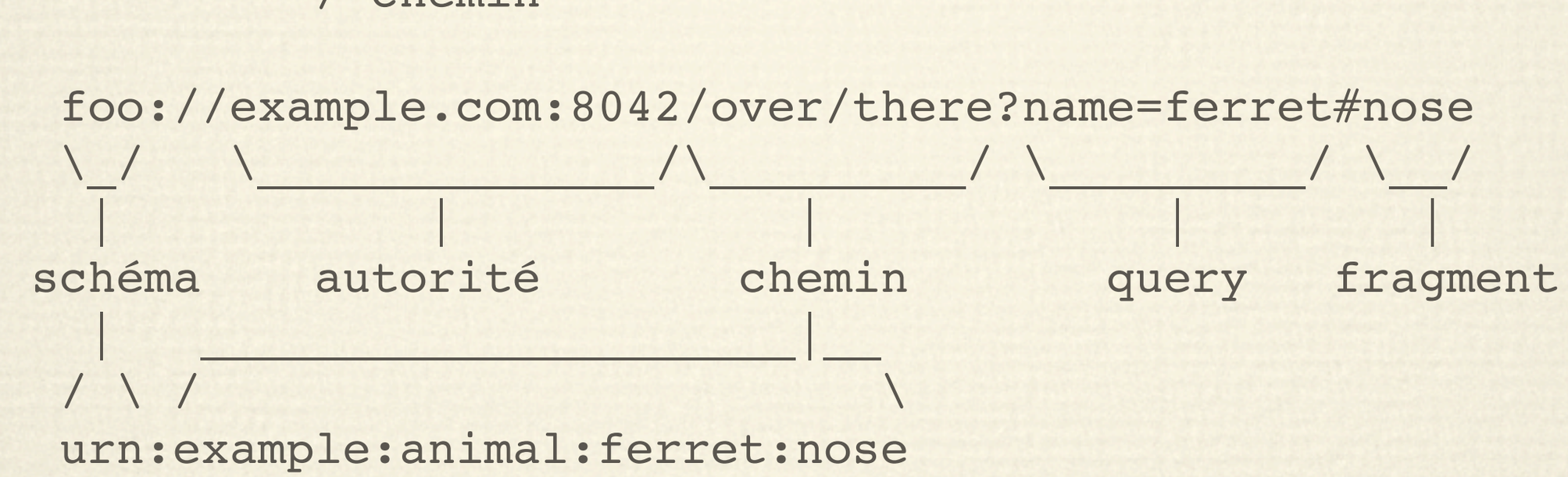
- Identifier les ressources de façon persistante
  `urn:isbn:1421610159`
  `urn:ietf:rfc:1097`
- Ne permet pas forcément de localiser...

Localiser : URL

- Permet de localiser et désigner une ressource : `http://site.org/qq/part/qq-chose`
- Si la ressource est déplacée, on ne la retrouve plus...

Identifier : URI

```
URI  =  schéma ":" adresse [ "?" query ] [ "#" fragment ]
   adresse = "//" autorité
                / chemin
```



Schéma

- Enregistrés auprès de l'IANA
- Quelques classiques :
  `data`, `file`, `ftp`, `gopher`, `http`, `https`, `imap`, `ldap`, `news`, `nntp`, `pop`, `rstp`, `tel`, `telnet`, `urn`, `xmpp`

Autorité

- `//autorite` : autorité qui définit la hiérarchie qui suit
- `autorite = [userinfo@]host[:port]`
- attention à la sécurité avec `userinfo` !
- `host` est un nom DNS ou une adresse IP

Chemin

- vide, absolu ou relatif
  `http://www.google.com`
  `http://www.site.org/images/bateau.jpg`
  `musique/chanson.mp3`

- chemin relatif : par rapport à une URI de base, pas d'autorité, directement le chemin.

Query

- Paramètres suivant le chemin
- `?nom=threepwood&prenom=guybrush`
- Typique des formulaires à action `GET`

Fragment

- Adressage secondaire par rapport à la ressource qui précède
- Typiquement une ancre dans un document HTML
  `http://www.gnu.org/#contactInfo`

En Python

- Bibliothèques **`urlparse`**, **`urllib`** et **`urllib2`**

```
urlparse.urlsplit(url)
urllib.splithost(url)
urllib.splitattr(url)
```

Domain Name System

DNS

- Les adresses IP c'est bon pour les machines
- Les humains préfèrent des noms compréhensibles :
  `209.85.129.147` → `www.google.com`
- Permet de changer d'IP de manière transparente

Espace de noms de domaines

- Structure hiérarchique
- Chaque espace est régi par un serveur de noms (*authoritative name server*)
- Un serveur de noms peut déléguer la responsabilité d'un sous-espace de noms à un sous-serveur

Entrées DNS (DNS records)

- (A) : adresse (IPv4)
- (AAAA) : adresse (IPv6)
- (NS) : name server
- (MX) : mail exchanger

Evolution

- À l'époque d'ARPAnet, un fichier `HOSTS.TXT` contenait la liste des correspondances nom-adresse
- 1983 : invention du DNS pour fonctionner sur les réseaux de grande taille
- RFC 882 et 883, remplacées ensuite par 1034 et 1035

Structure arborescente

(1) Racine des noms de domaine (*root*)

(2) Domaines de premier niveau (*top-level*) :
`.com`, `.fr`, `.org`, etc.

(3) Sous-domaines : `exemple.com`, puis
`www.exemple.com`, etc.

Contraintes

- Au plus 127 niveaux de sous-domaines
- Au plus 63 caractères par nom de niveau
- Au plus 253 caractères au total

International

- *Internationalized Domain Name* (IDNA)
- Noms de domaine avec caractères non-ASCII
- Encodage par *Punycode*
- 1998 : première implémentation
- 2009 : ICANN autorise la création d'IDNA sur les domaines de plus haut niveau

Résolution DNS

Pour trouver l'adresse d'une machine `www.exemple.com` :

(1) On interroge le serveur racine (`root`) qui renvoie l'adresse du serveur responsable du domaine `.com`

(2) On interroge le serveur `.com` qui renvoie l'adresse du serveur responsable de `exemple.com`

(3) On interroge le serveur de `exemple.com` qui nous donne l'adresse de la machine cherchée

Cache

- Inconvénient : le serveur racine est constamment interrogé
  - Chaque serveur se souvient des réponses obtenues pendant un certain temps (TTL)
  - Le cache provoque des décalages lorsque les informations sont modifiées

Initialisation

- Les requêtes DNS sont rarement envoyées directement par un utilisateur
- Les navigateurs web font les requêtes DNS automatiquement
- Le navigateur demande à l'OS l'adresse du premier serveur DNS
- l'OS est en général configuré sur le serveur DNS du FAI.

Résolution inverse

- On veut parfois connaître le nom associé à une adresse
- On utilise une requête DNS inverse :
  IP : `140.77.128.32`
  Nom : `32.128.77.140.in-addr.arpa.`
- La requête descend la hiérarchie comme dans le cas d'une résolution directe
- On utilise le suffixe `ip6.arpa` pour les IPv6

Types de données

Content-type

- Le protocole HTTP prévoit un champ Content-Type
- Exemples : `text/plain`, `text/html`, `image/jpeg`, `image/png`, `application/octet-stream`, `video/mpeg`, etc.
- Permet au navigateur de traiter les données

RFC et IANA

- Le problème du type de données est très répandu
- MIME Media Types `RFC 2045, 2046`
- Gérés par l'IANA
- Syntaxe : `type/sous-type`
- Types de base :
  `text`, `image`, `audio`, `video`, `application`, `multipart`, `message`

Caractères spéciaux

- En HTML, on a des entités pour écrire en US-ASCII sur 8 bits
- Problèmes évidents quand on utilise d'autres alphabets...
- Problème classique mal résolu :
  Charsets distincts

Encodage ?

- Spécifier l'encodage du texte
- Charsets IANA

```
Content-Type: text/html; charset=utf-8
Content-Type: text/html; charset=iso-8859-1
```

- Casse-têtes et transcodage à tout va

iso-latin-9

- L'ISO-8859-15 c'est en gros le latin-1 + '€'

ISO-8859-15																
	x0	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	l	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

http://en.wikipedia.org/wiki/ISO_8859-15

Unicode

- Table sur 32 bits “universelle”
- Recouvre la plupart des encodages 8 bits
- Encodages : utf-32, utf-16, utf-8
- Nouveau standard occidental : utf-8

Utilisation pratique

- Utiliser utf-8 comme codage
- Posséder un éditeur transcodeur
- Penser à déclarer son charset

Oui, mais...

- Comment deviner le Content-Type ?
  - par l'extension du fichier ?
  - par les méta-données du FS ?
  - en utilisant des *magic numbers* ?
- Il n'y a pas de solution miracle...

La magie des *magic numbers*

```
#PNG Image Format
   0x89504E470D0A1A0A → image/png
# JPEG images
   0xffd8 → image/jpeg
# Acrobat
   %PDF- → application/pdf
...
```

Et le texte ?

- C’est plus compliqué
- Problèmes d’encodage
- Problèmes de syntaxe libre

En pratique ?

- Déclarer le `Content-Type`
- Vérifier le `Content-Type` !
- Choisir un bon serveur, avec de bonnes heuristiques